
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

```
  SSSSSSSS  YY      YY   SSSSSSSS   SSSSSSSS  NN      NN  DDDDDDDD          JJ  BBBBBBBB      CCCCCCCC
  SSSSSSSS  YY      YY   SSSSSSSS   SSSSSSSS  NN      NN  DDDDDDDD          JJ  BBBBBBBB      CCCCCCCC
SS          YY      YY  SS          SS        NN      NN  DD      DD        JJ  BB      BB  CC
SS          YY      YY  SS          SS        NN      NN  DD      DD        JJ  BB      BB  CC
SS            YY  YY    SS          SS        NNNN    NN  DD      DD        JJ  BB      BB  CC
SS            YY  YY    SS          SS        NNNN    NN  DD      DD        JJ  BB      BB  CC
  SSSSSS        YY        SSSSSS      SSSSSS  NN  NN  NN  DD      DD        JJ  BBBBBBBB    CC
  SSSSSS        YY        SSSSSS      SSSSSS  NN  NN  NN  DD      DD        JJ  BBBBBBBB    CC
        SS      YY              SS          SS  NN    NNNN  DD      DD  JJ    JJ  BB      BB  CC
        SS      YY              SS          SS  NN    NNNN  DD      DD  JJ    JJ  BB      BB  CC
        SS      YY              SS          SS  NN      NN  DD      DD  JJ    JJ  BB      BB  CC                ....
        SS      YY              SS          SS  NN      NN  DD      DD  JJ    JJ  BB      BB  CC                ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD      JJJJJJ  BBBBBBBB      CCCCCCCC      ....
SSSSSSSS        YY      SSSSSSSS    SSSSSSSS    NN      NN  DDDDDDDD      JJJJJJ  BBBBBBBB      CCCCCCCC      ....

LL              IIIIII    SSSSSSSS
LL              IIIIII    SSSSSSSS
LL                II    SS
LL                II    SS
LL                II    SS
LL                II    SS
LL                II      SSSSSS
LL                II      SSSSSS
LL                II            SS
LL                II            SS
LL                II            SS
LL                II            SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  SYSSNDJBC - SEND MESSAGE TO JOB CONTROLLER
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY:
0000    31 ;       System services.
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module implements the Send to Job Controller ($SNDJBC) and the
0000    35 ;       Get Queue Information ($GETQUI) system services.
0000    36 ;
0000    37 ; AUTHOR: M. Jack, CREATION DATE: 29-Aug-1982
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       V03-011 JAK0218         J A Krycka      10-Jul-1984
0000    42 ;               Update tables to support new $SNDJBC and $GETQUI item codes.
0000    43 ;
0000    44 ;       V03-010 JAK0203         J A Krycka      17-Apr-1984
0000    45 ;               Update tables to support new $SNDJBC item codes.
0000    46 ;
0000    47 ;       V03-009 TMK0001         Todd M. Katz    04-Apr-1984
0000    48 ;               Re-write the action routine TRANSLATE_OBJECT to:
0000    49 ;
0000    50 ;                   1. Replace the recursive $TRNLOGs with $TRNLNMs.
0000    51 ;
0000    52 ;                   2. Eliminate the code that removes tabs, blanks, and null
0000    53 ;                      characters from names before attempting to translate them.
0000    54 ;                      Logical names should be handled in a systematic fashion
0000    55 ;                      throughout the system, and nobody else fiddles with them in
0000    56 ;                      such a fashion. However, after the recursive translations
0000    57 ;                      complete, at this time format the final translation according
```

```
0000     58 ;                  to the syntax expected for queue names. This involves
0000     59 ;                  removing tabs, null characters, and spaces from the final
0000     60 ;                  translation, and then upcasing it. This upcasing is done by
0000     61 ;                  means of the DEC multi-national character upcasing table.
0000     62 ;
0000     63 ;              3. Eliminate the code that upcases names before their
0000     64 ;                  translation because the $TRNLNMs will be done
0000     65 ;                  case-insensitive.
0000     66 ;
0000     67 ;              4. Micro-optimize the action routine.
0000     68 ;
0000     69 ;       V03-008 ACG0354         Andrew C. Goldstein,    13-Sep-1983
0000     70 ;               Change delete protection check to use alternate access
0000     71 ;               rather than access-granted.
0000     72 ;
0000     73 ;       V03-007 MLJ0118         Martin L. Jack, 22-Aug-1983
0000     74 ;               Guard against overlong resultant filename.  Update tables and
0000     75 ;               limits for new $GETQUI and $SNDJBC items.
0000     76 ;
0000     77 ;       V03-006 MLJ0115         Martin L. Jack, 30-Jul-1983
0000     78 ;               Changes for job controller baselevel.
0000     79 ;
0000     80 ;       V03-005 MLJ0114         Martin L. Jack, 23-Jun-1983
0000     81 ;               Add support for $GETQUI and for new $SNDJBC items.
0000     82 ;
0000     83 ;       V03-004 MLJ0112         Martin L. Jack, 28-Apr-1983
0000     84 ;               Update tables and limits for new items corresponding to job
0000     85 ;               controller baselevel.
0000     86 ;
C000     87 ;       V03-003 CWH1002         CW Hobbs        24-Feb-1983
0000     88 ;               Send extended pid and owner fields to the job controller.
0000     89 ;
0000     90 ;       V03-002 MLJ0106         Martin L. Jack, 1-Mar-1983
0000     91 ;               Update tables and limits for new items corresponding to job
0000     92 ;               controller baselevel.
0000     93 ;
0000     94 ;       V03-001 MLJ0103         Martin L. Jack, 7-Jan-1983
0000     95 ;               Update tables and limits for new items corresponding to job
0000     96 ;               controller baselevel.
0000     97 ;
0000     98 ;--
```

```
                        0000   100          .SBTTL  DATA DEFINITIONS
                        0000   101
                        0000   102 ;
                        0000   103 ; EXTERNAL SYMBOLS:
                        0000   104 ;
                        0000   105
                        0000   106          $ACBDEF                 ; Define AST control block offsets
                        0000   107          $ACMDEF                 ; Define accounting manager offsets
                        0000   108          $ARMDEF                 ; Define access rights mask
                        0000   109          $ATRDEF                 ; Define ACP attribute codes
                        0000   110          $FABDEF                 ; Define RMS file attributes block offsets
                        0000   111          $FATDEF                 ; Define RMS file attribute area offsets
                        0000   112          $FIBDEF                 ; Define file information block offsets
                        0000   113          $IODEF                  ; Define I/O function codes
                        0000   114          $LNMDEF                 ; Define logical name system service symbols
                        0000   115          $MSGDEF                 ; Define mailbox message types
                        0000   116          $NAMDEF                 ; Define RMS name block offsets
                        0000   117          $PCBDEF                 ; Define process control block offsets
                        0000   118          $PHDDEF                 ; Define process header offsets
                        0000   119          $PSLDEF                 ; Define processor status longword offsets
                        0000   120          $QUIDEF                 ; Define $GETQUI function and item codes
                        0000   121          $SJCDEF                 ; Define $SNDJBC function and item codes
                        0000   122          $SSDEF                  ; Define system status codes
                        0000   123
                        0000   124 ;
                        0000   125 ; LOCAL SYMBOLS:
                        0000   126 ;
                        0000   127 ; Extension to AST control block.  These definitions are also known to JOBCTL.
                        0000   128 ;
                        0000   129
                        0000   130          $DEFINI ACB
                        0000   131          .=ACB$L_KAST+4                  ; Position to end
                        001C   132 $DEF     ACB_L_IMGCNT    .BLKL   1       ; Image counter
                        0020   133 $DEF     ACB_L_EFN       .BLKL   1       ; Event flag number
                        0024   134 $DEF     ACB_L_IOSB      .BLKL   1       ; IOSB address
                        0028   135 $DEF     ACB_L_STATUS    .BLKL   1       ; Status for IOSB
                        002C   136 $DEF     ACB_W_ITEMCOUNT .BLKW   1       ; Count of data items
                        002E   137 $DEF     ACB_B_ITEMS     .BLKB   0       ; Base of item descriptors
                        002E   138          $DEFEND ACB
                        0000   139
                        0000   140 ;
                        0000   141 ; Argument list offsets (based on AP).
                        0000   142 ;
                        0000   143
00000004                0000   144 EFN=          4                          ; Event flag number
00000008                0000   145 FUNC=         8                          ; Function code
0000000C                0000   146 NULARG=       12                         ; Reserved argument
00000010                0000   147 ITMLST=       16                         ; Address of item descriptor list
00000014                0000   148 IOSB=         20                         ; Address of I/O status block
00000018                0000   149 ASTADR=       24                         ; Address of AST routine
0000001C                0000   150 ASTPRM=       28                         ; AST parameter
                        0000   151
                        0000   152 ;
                        0000   153 ; Data table offsets (based on R11).
                        0000   154 ;
                        0000   155
00000000                0000   156 MSG_CODE=     0                          ; Message function code
```

```
00000004  0000   157 MAX_FUNC=        4                       ; Highest function code minus one
00000008  0000   158 MAX_ITEM=        8                       ; Highest item code minus one
0000000C  0000   159 BOOL_ITEM=       12                      ; Pointer to boolean item table
00000010  0000   160 OUTPUT_ITEM=     16                      ; Pointer to output item table
00000014  0000   161 SPECIAL_TABLE=   20                      ; Pointer to special handling table
          0000   162
          0000   163 ;
          0000   164 ; Fixed work area offsets (above FP).
          0000   165 ;
          0000   166
FFFFFFFC  0000   167 FILE_ID=         -4                      ; Pointer to file identification item
FFFFFFF8  0000   168 FLAGS=           -8                      ; Pointer to miscellaneous status flags
00000000  0000   169 DELETE_FLAG=     0                       ; SJC$_DELETE_FILE seen
          0000   170
          0000   171 ;
          0000   172 ; Miscellaneous definitions.
          0000   173 ;
          0000   174
0000005A  0000   175 FIXED_AREA=      ACM$Q_SYSTIME+8 + 22    ; Size of fixed area of message
```

```
                     0000   177
                     0000   178 ;
                     0000   179 ; LOCAL STORAGE:
                     0000   180 ;
                     0000   181 ; This table is indexed by item code (normalized to zero origin).  It
                     0000   182 ; identifies items classified as Boolean.
                     0000   183 ;
                     0000   184
                00000000    185          .PSECT  Y$EXEPAGED
                     0000   186 SNDJBC_BOOL_ITEM:
99EAA198             0000   187          .LONG   ^B10011001111101010101000110011000
B7005D3B             0004   188          .LONG   ^B10110111000000000101110100111011
EB751BDF             0008   189          .LONG   ^B11101011011101010001101111011111
C03B5DFC             000C   190          .LONG   ^B11000000001110110101110111111100
0A9F3F09             0010   191          .LONG   ^B00001010100111111001111110000 1001
0000001E             0014   192          .LONG   ^B00000000000000000000000000011110
00000000             0018   193          .LONG   ^B00000000000000000000000000000000
00000000             001C   194          .LONG   ^B00000000000000000000000000000000
                     0020   195 GETQUI_BOOL_ITEM:
00000000             0020   196          .LONG   ^B00000000000000000000000000000000
00000000             0024   197          .LONG   ^B00000000000000000000000000000000
00300000             0028   198          .LONG   ^B00000000001100000000000000000000
00000000             002C   199          .LONG   ^B00000000000000000000000000000000
00000000             0030   200          .LONG   ^B00000000000000000000000000000000
00000000             0034   201          .LONG   ^B00000000000000000000000000000000
00000000             0038   202          .LONG   ^B00000000000000000000000000000000
00000000             003C   203          .LONG   ^B00000000000000000000000000000000
                     0040   204
                     0040   205 ;
                     0040   206 ; This table is indexed by item code (normalized to zero origin).  It
                     0040   207 ; identifies items classified as output.
                     0040   208 ;
                     0040   209
                     0040   210 SNDJBC_OUTPUT_ITEM:
40000400             0040   211          .LONG   ^B01000000000000000000010000000000
00000000             0044   212          .LONG   ^B00000000000000000000000000000000
00800000             0048   213          .LONG   ^B00000000100000000000000000000000
00000000             004C   214          .LONG   ^B00000000000000000000000000000000
00000000             0050   215          .LONG   ^B00000000000000000000000000000000
00000600             0054   216          .LONG   ^B00000000000000000000011000000000
00000000             0058   217          .LONG   ^B00000000000000000000000000000000
00000000             005C   218          .LONG   ^B00000000000000000000000000000000
                     0060   219 GETQUI_OUTPUT_ITEM:
FFFFFFFF             0060   220          .LONG   ^B11111111111111111111111111111111
FFFFFFFF             0064   221          .LONG   ^B11111111111111111111111111111111
3F0FC7FF             0068   222          .LONG   ^B00111111000011111100011111111111
00000000             006C   223          .LONG   ^B00000000000000000000000000000000
00000000             0070   224          .LONG   ^B00000000000000000000000000000000
00000000             0074   225          .LONG   ^B00000000000000000000000000000000
00000000             0078   226          .LONG   ^B00000000000000000000000000000000
00000000             007C   227          .LONG   ^B00000000000000000000000000000000
                     0080   228
                     0080   229 ;
                     0080   230 ; This table identifies item codes that require special translation and the
                     0080   231 ; routine that performs the translation.
                     0080   232 ;
                     0080   233
```

```
                 0080   234 SNDJBC_SPECIAL_TABLE:
            000C 0080   235         .WORD   SJC$_CHARACTERISTIC_NAME
00000480'   C082   236         .LONG   TRANSLATE_OBJECT
            001A 0086   237         .WORD   SJC$_DESTINATION_QUEUE
00000480'   0088   238         .LONG   TRANSLATE_OBJECT
            0027 008C   239         .WORD   SJC$_FILE_IDENTIFICATION
00000369'   008E   240         .LONG   FILE_IDENTIFICATION
            002A 0092   241         .WORD   SJC$_FILE_SPECIFICATION
000002F1'   0094   242         .LONG   FILE_SPECIFICATION
            0036 0098   243         .WORD   SJC$_FORM_NAME
00000480'   009A   244         .LONG   TRANSLATE_OBJECT
            0046 009E   245         .WORD   SJC$_GENERIC_TARGET
00000480'   00A0   246         .LONG   TRANSLATE_OBJECT
            0061 00A4   247         .WORD   SJC$_LOG_QUEUE
00000480'   00A6   248         .LONG   TRANSLATE_OBJECT
            0086 00AA   249         .WORD   SJC$_QUEUE
00000480'   00AC   250         .LONG   TRANSLATE_OBJECT
            0000 00B0   251         .WORD   0
                 00B2   252 GETQUI_SPECIAL_TABLE:
            004D 00B2   253         .WORD   QUI$_SEARCH_NAME
00000480'   00B4   254         .LONG   TRANSLATE_OBJECT
            0000 00B8   255         .WORD   0
                 00BA   256
                 00BA   257 ;
                 00BA   258 ; The following values are needed as arguements to the $TRNLNMs performed by
                 00BA   259 ; the action routine TRANSLATE_OBJECT.
                 00BA   260 ;
                 00BA   261
                 00BA   262 TRNLNM_ATTR:                         ; Optional attributes for $TRNLNMs
            02000000 00BA   263         .LONG   LNM$M_CASE_BLIND    ; Translations are done case-insensitive
                 00BE   264
                 00BE   265 TRNLNM_TABLE:                        ; Tables in which to do the translations
49 46 24 4D 4E 4C 000000C6'010E0000' 00BE   266         .ASCID  /LNM$FILE_DEV/
                  56 45 44 5F 45 4C  00CC
```

```
                        00D2   268          .SBTTL  EXE$SNDJBC - Send message to job controller
                        00D2   269
                        00D2   270 ;++
                        00D2   271 ;
                        00D2   272 ; EXE$SNDJBC - SEND MESSAGE TO JOB CONTROLLER
                        00D2   273 ; EXE$GETQUI - GET QUEUE INFORMATION
                        00D2   274 ;
                        00D2   275 ; FUNCTIONAL DESCRIPTION:
                        00D2   276 ;
                        00D2   277 ;       This routine provides the send to job controller and get queue
                        00D2   278 ;       information system services.  The action is to build a message from the
                        00D2   279 ;       user's input data and send it to the job controller mailbox.  At request
                        00D2   280 ;       completion, the job controller queues a special kernel AST to routine
                        00D2   281 ;       EXE$JBCRSP to return status to this process.
                        00D2   282 ;
                        00D2   283 ; INPUTS:
                        00D2   284 ;       EFN(AP)          = Number of the event flag to set at request completion
                        00D2   285 ;       FUNC(AP)         = Function code
                        00D2   286 ;       NULARG(AP)       = Reserved argument, must be zero
                        00D2   287 ;       ITMLST(AP)       = Address of a list of item descriptors
                        00D2   288 ;       IOSB(AP)         = Address of a quadword to receive completion status
                        00D2   289 ;       ASTADR(AP)       = Address of an AST routine to be called at request
                        00D2   290 ;                          completion
                        00D2   291 ;       ASTPRM(AP)       = Longword AST parameter
                        00D2   292 ;
                        00D2   293 ; OUTPUTS:
                        00D2   294 ;       R0               = Status of the operation
                        00D2   295 ;
                        00D2   296 ; STATUS CODES RETURNED:
                        00D2   297 ;       SS$_NORMAL       Successful operation
                        00D2   298 ;
                        00D2   299 ;       SS$_ACCVIO       Unable to write IOSB, read ITMLST, read or write item
                        00D2   300 ;                        buffer, write return length buffer
                        00D2   301 ;       SS$_BADPARAM     Invalid FUNC, nonzero NULARG, invalid item code,
                        00D2   302 ;                        invalid zero or nonzero field in item descriptor
                        00D2   303 ;       SS$_DEVOFFLINE   No job controller
                        00D2   304 ;       SS$_EXASTLM      Exceeded ASTLM quota
                        00D2   305 ;       SS$_ILLEFC       Illegal event flag number
                        00D2   306 ;       SS$_INSFMEM      Insufficient system memory to complete request
                        00D2   307 ;       SS$_MBFULL       Job controller mailbox full
                        00D2   308 ;       SS$_MBTOOSML     Message too large for job controller mailbox
                        00D2   309 ;       SS$_UNASEFC      Unassociated event flag cluster
                        00D2   310 ;--
                        00D2   311
                        00D2   312 GETQUI_DATA:
00000010                00D2   313          .LONG   MSG$_GETQUI                ; Message function code
00000008                00D6   314          .LONG   QUI$_RESERVED_FUNC_2-1     ; Highest function code minus one
0000005D                00DA   315          .LONG   QUI$_RESERVED_OUTPUT_6-1   ; Highest item code minus one
00000020'               00DE   316          .LONG   GETQUI_BOOL_ITEM           ; Pointer to boolean item table
00000060'               00E2   317          .LONG   GETQUI_OUTPUT_ITEM         ; Pointer to output item table
000000B2'               00E6   318          .LONG   GETQUI_SPECIAL_TABLE       ; Pointer to special handling table
                        00EA   319
                        00EA   320 SNDJBC_DATA:
0000000F                00EA   321          .LONG   MSG$_SNDJBC                ; Message function code
0000001F                00EE   322          .LONG   SJC$_RESERVED_FUNC_2-1     ; Highest function code minus one
000000AA                00F2   323          .LONG   SJC$_RESERVED_OUTPUT_2-1   ; Highest item code minus one
00000000'               00F6   324          .LONG   SNDJBC_BOOL_ITEM           ; Pointer to boolean item table
```

```
                00000040'  00FA   325          .LONG   SNDJBC_OUTPUT_ITEM      ; Pointer to output item table
                00000080'  00FE   326          .LONG   SNDJBC_SPECIAL_TABLE    ; Pointer to special handling table
                           0102   327
                           0102   328          .ENABL  LSB
                           0102   329
                           0102   330 ACCVIO:
                     50    0C  D0  0102   331          MOVL    #SS$_ACCVIO,R0          ; Set access violation status
                           0A  11  0105   332          BRB     10$
                           0107   333
                           0107   334 BADPARAM:
                     50    14  D0  0107   335          MOVL    #SS$_BADPARAM,R0        ; Set bad parameter status
                           05  11  010A   336          BRB     10$
                           010C   337
                           010C   338 INSFMEM:
                  50 0124  8F  3C  010C   339          MOVZWL  #SS$_INSFMEM,R0         ; Set insufficient memory status
                     0472  31  0111   340 10$:       BRW     ERROR
                           0114   341
                           0114   342
                           0114   343 EXE$GETQUI::                               ; Get queue information
                       0FFC        0114   344          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                  5B B9 AF  9E  0116   345          MOVAB   GETQUI_DATA,R11         ; Point to $GETQUI data table
                           06  11  011A   346          BRB     20$                     ; Join common code
                           011C   347
                           011C   348
                           011C   349 EXE$SNDJBC::                               ; Send to job controller
                       0FFC        011C   350          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                  5B C9 AF  9E  011E   351          MOVAB   SNDJBC_DATA,R11         ; Point to $SNDJBC data table
                           0122   352
                           0122   353 ;
                           0122   354 ; Point to lowest usable stack address.
                           0122   355 ;
                           0122   356
                     7E    7C  0122   357 20$:       CLRQ    -(SP)                   ; Allocate fixed work area
                     50    DC  0124   358          MOVPSL  R0                      ; Get PSL
                  50 50    02  18  EF  0126   359          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,R0,R0 ; Get current mode
00000000'EF40      0000008E  8F  C1  012B   360          ADDL3   #FIXED_AREA+52, -       ; Allow slop for fixed message area plus
                     5A        0137   361                  CTL$AL_STACKLIM[R0], -  ; 52 bytes for $CMKRNL frame and
                           0138   362                  R10                     ; parameters
                           0138   363
                           0138   364 ;
                           0138   365 ; Check for and clear I/O status block.
                           0138   366 ;
                           0138   367
                  50 14 AC  D0  0138   368          MOVL    IOSB(AP),R0             ; Get IOSB address
                        08  13  013C   369          BEQL    30$                     ; Branch if none
                           013E   370          IFNOWRT #8,(R0),ACCVIO          ; Check write access to IOSB
                        60  7C  0144   371          CLRQ    (R0)                    ; Clear IOSB
                           0146   372
                           0146   373 ;
                           0146   374 ; Validate function code.
                           0146   375 ;
                           0146   376
                  52 08 AC  01  C3  0146   377 30$:       SUBL3   #1,FUNC(AP),R2          ; Get function code and subtract out
                           014B   378                                          ; smallest value to get zero origin
                  04 AB    52  D1  014B   379          CMPL    R2,MAX_FUNC(R11)        ; Check against largest value
                           B6  1A  014F   380          BGTRU   BADPARAM                ; Branch if invalid value
                           0151   381
```

```
                        0151   382 ;
                        0151   383 ; Validate unused argument (must be zero).
                        0151   384 ;
                        0151   385
                OC AC   D5 0151   386         TSTL    NULARG(AP)              ; Unused argument zero?
                   B1   12 0154   387         BNEQ    BADPARAM                ; Branch if not zero
                        0156   388
                        0156   389 ;
                        0156   390 ; Process the item list to build the job controller message.  During this loop:
                        0156   391 ;
                        0156   392 ;       R5  = buffer size
                        0156   393 ;       R6  = item code
                        0156   394 ;       R7  = buffer address
                        0156   395 ;       R8  = return length address
                        0156   396 ;       R9  = pointer to item list
                        0156   397 ;       R10 = pointer to lowest available stack address
                        0156   398 ;       R11 = pointer to service-specific data area
                        0156   399 ;
                        0156   400
            59    10 AC   D0 0156   401         MOVL    ITMLST(AP),R9           ; Get item descriptor list address
                      0E   13 015A   402         BEQL    40$                     ; Branch if no item list
                           015C   403         IFNORD  #4,(R9),90$             ; Check read access to first longword
                55    89   3C 0162   404 ITEM:  MOVZWL  (R9)+,R5               ; Get buffer size value
                56    89   3C 0165   405         MOVZWL  (R9)+,R6                ; Get item code value
                      03   12 0168   406         BNEQ    50$                     ; Branch if nonzero, list not ended
                   00B0    31 016A   407 40$:    BRW     FINISH_MESSAGE          ; Branch if zero, list ended
          52    56    01   C3 016D   408 50$:    SUBL3   #1,R6,R2                ; Subtract out smallest value to get
                           0171   409                                           ; zero origin
            08 AB    52   D1 0171   410         CMPL    R2,MAX_ITEM(R11)        ; Check against largest value
                     59   1A 0175   411         BGTRU   100$                    ; Branch if invalid value
                           0177   412         IFNORD  #12,(R9),90$            ; Check read access to second and third
                           017D   413                                           ; longwords of this item and first
                           017D   414                                           ; longword of next item
                57   89   7D 017D   415         MOVQ    (R9)+,R7                ; Get buffer address and return length
                           0180   416                                           ; address
                           0180   417 ;
                           0180   418 ; Boolean item.  Store the item code.
                           0180   419 ;
                           0180   420
          17 0C BB   52   E1 0180   421         BBC     R2,@BOOL_ITEM(R11),60$  ; Branch if not boolean item
           50    5E   02   C3 0185   422         SUBL3   #2,SP,R0                ; Get lowest address that will be used
                 5A   50   D1 0189   423         CMPL    R0,R10                  ; Compare against that available
                      45   1F 018C   424         BLSSU   110$                    ; Branch if space exceeded
                7E   56   B0 018E   425         MOVW    R6,-(SP)                ; Store item code
                18   56   B1 0191   426         CMPW    R6,#SJC$_DELETE_FILE    ; Check for file deletion
                     CC   12 0194   427         BNEQ    ITEM
            F8 AD    01   88 0196   428         BISB    #1@DELETE_FLAG,FLAGS(FP); Note file deletion for postprocessing
                     C6   11 019A   429         BRB     ITEM                    ; Branch to process next item
                           019C   430
                           019C   431 ;
                           019C   432 ; Input or output item.  Set up to call EXE$PROBEx.
                           019C   433 ;
                           019C   434
                50   57   D0 019C   435 60$:    MOVL    R7,R0                   ; R0 = buffer address
                51   55   D0 019F   436         MOVL    R5,R1                   ; R1 = buffer length
                     53   D4 01A2   437         CLRL    R3                      ; R3 = probe against previous mode
                     55   DD 01A4   438         PUSHL   R5                      ; Save R5 across call
```

```
                                                M 11
                     01A6   439
                     01A6   440 ;
                     01A6   441 ; Input item.  Ensure that the buffer is accessible.
                     01A6   442 ;
                     01A6   443
47 10 BB    52   E0 01A6   444          BBS     R2,@OUTPUT_ITEM(R11),120$ ; Branch if output item
   00000000'EF   16 01AB   445          JSB     EXE$PROBER                ; Probe read access to buffer
         55 8ED0    01B1   446          POPL    R5                        ; Restore R5
      16 50   E9    01B4   447          BLBC    R0,90$                    ; Branch if no access
                     01B7   448
                     01B7   449 ;
                     01B7   450 ; Test for items that receive special translation.
                     01B7   451 ;
                     01B7   452
   50   14 AB   D0 01B7   453          MOVL    SPECIAL_TABLE(R11),R0     ; Point to special handling table
      51   60   3C 01BB   454 70$:     MOVZWL  (R0),R1                   ; Pick up item code and test if ended
           16   13 01BE   455          BEQL    INPUT_ITEM                ; Branch if ended
      51   56   D1 01C0   456          CMPL    R6,R1                     ; Correct item code?
           03   12 01C3   457          BNEQ    80$                       ; Branch if not
        02 B0   17 01C5   458          JMP     @2(R0)                    ; Jump to processing routine
      50   06   C0 01C8   459 80$:     ADDL2   #6,R0                     ; Increment to next table entry
           EE   11 01CB   460          BRB     70$                       ; Loop to compare next
                     01CD   461
                     01CD   462 ;
                     01CD   463 ; Helper branches.
                     01CD   464 ;
                     01CD   465
         FF32   31 01CD   466 90$:     BRW     ACCVIO
         FF34   31 01D0   467 100$:    BRW     BADPARAM
         FF36   31 01D3   468 110$:    BRW     INSFMEM
                     01D6   469
                     01D6   470 ;
                     01D6   471 ; Ordinary input item.  Store the item code, buffer length, and contents.
                     01D6   472 ;
                     01D6   473
                     01D6   474 INPUT_ITEM:
   50   5E   55   C3 01D6   475          SUBL3   R5,SP,R0                  ; Get lowest address that will be used
        50   04   C2 01DA   476          SUBL2   #4,R0                     ;
        5A   50   D1 01DD   477          CMPL    R0,R10                    ; Compare against that available
             F1   1F 01E0   478          BLSSU   110$                      ; Branch if space exceeded
        5E   50   D0 01E2   479          MOVL    R0,SP                     ; Allocate the space
        80   56   B0 01E5   480          MOVW    R6,(R0)+                  ; Store item code
        80   55   B0 01E8   481          MOVW    R5,(R0)+                  ; Store item length
   60   67   55   28 01EB   482          MOVC3   R5,(R7),(R0)              ; Store item value
           FF70   31 01EF   483          BRW     ITEM                      ; Branch to process next item
                     01F2   484
                     01F2   485 ;
                     01F2   486 ; Output item.  Ensure that the buffer, and return length if specified, are
                     01F2   487 ; accessible, and store the item code, buffer length, buffer address, and return
                     01F2   488 ; length address.
                     01F2   489 ;
                     01F2   490
   00000000'EF   16 01F2   491 120$:    JSB     EXE$PROBEW                ; Probe write access to buffer
         55 8ED0    01F8   492          POPL    R5                        ; Restore R5
      CF 50   E9    01FB   493          BLBC    R0,90$                    ; Branch if no access
         58   D5 01FE   494          TSTL    R8                        ; Test if return length specified
           06   13 0200   495          BEQL    130$                      ; Branch if not specified
```

```
                                                N 11
                     0202    496                IFNOWRT #2,(R8),90$                     ; Probe write access to length word
 50  5E   0C   C3     0208    497 130$:          SUBL3   #12,SP,R0                       ; Get lowest address that will be used
     5A   50   D1     020C    498                CMPL    R0,R10                          ; Compare against that available
          C2   1F     020F    499                BLSSU   110$                            ; Branch if space exceeded
     7E   57   7D     0211    500                MOVQ    R7,-(SP)                        ; Store item buffer addresses
     7E   55   B0     0214    501                MOVW    R5,-(SP)                        ; Store item length
     7E   56   B0     0217    502                MOVW    R6,-(SP)                        ; Store item code
        FF45   31     021A    503                BRW     ITEM                            ; Branch to process next item
                     021D    504
                     021D    505 ;
                     021D    506 ; To here when all items have been processed. Do necessary postprocessing
                     021D    507 ; and finish the message.
                     021D    508 ;
                     021D    509
                     021D    510 FINISH_MESSAGE:
 5A  FC AD   D0      021D    511                MOVL    FILE_ID(FP),R10                 ; Get file ID item, if any
          03 13      0221    512                BEQL    140$                            ; Branch if none
        017C 30      0223    513                BSBW    POSTPROCESS_FID                 ; Deal with it
                     0226    514
                     0226    515 ;
                     0226    516 ; Build the message header.
                     0226    517 ;
                     0226    518
56  00000000'9F      D0      0226    519 140$:          MOVL    @#CTL$GL_PCB,R6         ; Get PCB address
57  00000000'9F      D0      022D    520                MOVL    @#CTL$GL_PHD,R7         ; Get PHD address
     7E   08 AC      B0      0234    521                MOVW    FUNC(AP),-(SP)          ; Store function code
     7E   18 AC      7D      0238    522                MOVQ    ASTADR(AP),-(SP)        ; Store AST address and parameter
          14 AC      DD      023C    523                PUSHL   IOSB(AP)                ; Store IOSB address
     7E   04 AC      9A      023F    524                MOVZBL  EFN(AP),-(SP)           ; Store event flag number
        00F4 C7      DD      0243    525                PUSHL   PHD$L_IMGCNT(R7)        ; Store image counter
          5E   08    C2      0247    526                SUBL2   #8,SP                   ; Make space for system time
 6E   00000000'EF    7D      024A    527 150$:          MOVQ    EXE$GQ_SYSTIME,(SP)     ; Store current time
 6E   00000000'EF    D1      0251    528                CMPL    EXE$GQ_SYSTIME,(SP)     ; Verify that value acquired was not
          F0 12      0258    529                BNEQ    150$                            ;  being modified at the same time
04 AE  00000004'EF   D1      025A    530                CMPL    EXE$GQ_SYSTIME+4,4(SP)  ;  and store it again if it changed
          E6 12      0262    531                BNEQ    150$                            ;
     7E   44 A6      7D      0264    532                MOVQ    PCB$T_TERMINAL(R6),-(SP); Store terminal name
          68 A6      DD      0268    533                PUSHL   PCB$L_EOWNER(R6)        ; Store extended owner process ID
          24 A6      DD      026B    534                PUSHL   PCB$L_STS(R6)           ; Store process status
          64 A6      DD      026E    535                PUSHL   PCB$L_EPID(R6)          ; Store extended process ID
             7E      B4      0271    536                CLRW    -(SP)                   ; Clear spare word
             50      DC      0273    537                MOVPSL  R0                      ; Get PSL
 50  50  02          16      EF      0275    538        EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
     7E              50      90      027A    539        MOVB    R0,-(SP)                ; Store requester's mode
  7E  1F  2F         A6      83      027D    540        SUBB3   PCB$B_PRIB(R6),#31,-(SP); Store base priority
          5E         14      C2      0282    541        SUBL2   #20,SP                  ; Allocate space for next field
6E  00000000'9F      14      28      0285    542        MOVC3   #20,@#CTL$T_USERNAME,(SP) ; Store username and account name
        00BC C6      DD      028D    543                PUSHL   PCB$L_UIC(R6)           ; Store UIC
          7E   67    7D      0291    544                MOVQ    PHD$Q_PRIVMSK(R7),-(SP) ; Store privileges
          7E   68    3C      0294    545                MOVZWL  MSG_CODE(R11),-(SP)     ; Store message type, clear mailbox
                     0297    546
                     0297    547 ;
                     0297    548 ; Finished building the message.  Push the address of the service argument
                     0297    549 ; list, and the address and length of the message, and enter kernel mode to
                     0297    550 ; complete argument list processing and write the message.
                     0297    551 ;
                     0297    552
```

```
                                                B 12
                     5E   DD  0297   553          PUSHL   SP                      ; Push address of message
                7E   5D   6E  C3  0299   554          SUBL3   (SP),FP,-(SP)           ; Push length of message
                     6E   08  C2  029D   555          SUBL    #8,(SP)                ; Deduct fixed work area
                     5C   DD  02A0   556          PUSHL   AP                      ; Push service argument list
                                  02A2   557          $CMKRNL_S B^170$,(SP)          ; Finish in kernel mode
                   01 50  E9  02AE   558          BLBC    R0,160$                ; Branch if error
                          04  02B1   559          RET                            ; Return
                  02D1    31  02B2   560 160$:    BRW     ERROR                  ; Helper branch
                                  02B5   561
                                  02B5   562 ;
                                  02B5   563 ; Kernel mode routine to finish processing.
                                  02B5   564 ;
                                  02B5   565
                  007C            02B5   566 170$:    .WORD   ^M<R2,R3,R4,R5,R6>     ; Entry mask
                                  02B7   567
                                  02B7   568 ;
                                  02B7   569 ; Get parameter list address and PCB address.
                                  02B7   570 ;
                                  02B7   571
              54  00000000'9F     D0  02B7   572          MOVL    @#CTL$GL_PCB,R4        ; Get PCB address
                     56   8C     D0  02BE   573          MOVL    (AP)+,R6               ; Get service parameter list address
                                  02C1   574
                                  02C1   575 ;
                                  02C1   576 ; Clear event flag.
                                  02C1   577 ;
                                  02C1   578
                    53   04 A6    9A  02C1   579          MOVZBL  EFN(R6),R3             ; Get event flag number
                00000000'EF       16  02C5   580          JSB     SCH$CLREF              ; Clear this event flag
                      22 50       E9  02CB   581          BLBC    R0,190$                ; Return on errors
                                  02CE   582
                                  02CE   583 ;
                                  02CE   584 ; Check and charge AST quota.
                                  02CE   585 ;
                                  02CE   586
                    18 A6     D5  02CE   587          TSTL    ASTADR(R6)             ; AST routine specified?
                       0D     13  02D1   588          BEQL    180$                   ; Branch if none
                50  2A04 8F     3C  02D3   589          MOVZWL  #SS$_EXASTLM,R0        ; Assume AST quota exceeded status
                    38 A4     B5  02D8   590          TSTW    PCB$W_ASTCNT(R4)       ; Test for quota exceeded
                       13     15  02DB   591          BLEQ    190$                   ; Branch if exceeded
                    38 A4     B7  02DD   592          DECW    PCB$W_ASTCNT(R4)       ; Charge AST quota
                                  02E0   593
                                  02E0   594 ;
                                  02E0   595 ; Send the message.
                                  02E0   596 ;
                                  02E0   597
                    53   6C     7D  02E0   598 180$:    MOVQ    (AP),R3                ; R3 = size, R4 = address of message
              55  00000000'EF     9E  02E3   599          MOVAB   SYS$GL_JOBCTLMB,R5     ; R5 = mailbox UCB address
                00000000'EF       16  02EA   600          JSB     EXE$SENDMSG            ; Send message
                                  04  02F0   601 190$:    RET                            ; Return
                                  02F1   602          .DSABL  LSB
```

```
                     02F1    604
                     02F1    605 ;
                     02F1    606 ; Stack work area offsets for next routine.
                     02F1    607 ;
                     02F1    608
00000000             02F1    609 FWA_DVI=        0                       ; DVI
00000010             02F1    610 FWA_FID=        16                      ; FID
00000016             02F1    611 FWA_DID=        22                      ; DID
0000001C             02F1    612 FWA_FILE_SIZE=  28                      ; File size in blocks
                     02F1    613                                         ; (spare longword)
00000024             02F1    614 FWA_FILE_SPEC=  36                      ; File specification
00000124             02F1    615 FWA_RECATTR=    292                     ; Record attributes
00000144             02F1    616 FWA_CHAN=       324                     ; Channel assigned to device
00000148             02F1    617 FWA_IOSB=       328                     ; I/O status block
                     02F1    618
0000001C             02F1    619 FWA_FAB=        28                      ; FAB block
0000006C             02F1    620 FWA_NAM=        FWA_FAB + FAB$C_BLN     ; NAM block
000000CC             02F1    621 FWA_ESA=        FWA_NAM + NAM$C_BLN     ; Expanded string
000001CB             02F1    622 FWA_SIZE=       FWA_ESA + NAM$C_MAXRSS  ; Length of area
                     02F1    623
00000024             02F1    624 FWA_DVI_DESC=   36                      ; Descriptor for device name
                     02F1    625
00000024             02F1    626 FWA_FIB_DESC=   36                      ; Descriptor for FIB
0000002C             02F1    627 FWA_FIB=        44                      ; File information block
0000006C             02F1    628 FWA_ATRLIST=    FWA_FIB+FIB$C_LENGTH    ; Attribute list
                     02F1    629
                     02F1    630         .ENABL  LSB
                     02F1    631 FILE_SPECIFICATION:                     ; Translate SJC$_FILE_SPECIFICATION
                     02F1    632
                     02F1    633 ;
                     02F1    634 ;       R5  = buffer size
                     02F1    635 ;       R6  = item code
                     02F1    636 ;       R7  = buffer address
                     02F1    637 ;       R10 = pointer to lowest available stack address
                     02F1    638 ;
                     02F1    639 ; Check that the parameter is the correct length.
                     02F1    640 ;
                     02F1    641
         FC AD   D5  02F1    642         TSTL    FILE_ID(FP)             ; See if there is already a filespec
            6D   12  02F4    643         BNEQ    20$                     ; Branch if so
   00FF  8F   55   B1  02F6    644         CMPW    R5,#255                 ; Ensure no longer than 255 bytes
            66   1A  02FB    645         BGTRU   20$                     ; Branch if incorrect
                     02FD    646
                     02FD    647 ;
                     02FD    648 ; Check for sufficient space to allocate the work area, and do so.
                     02FD    649 ;
                     02FD    650
   50    FE35 CE   9E  02FD    651         MOVAB   -FWA_SIZE(SP),R0        ; Get lowest address that will be used
         5A   50   D1  0302    652         CMPL    R0,R10                  ; Compare against that available
              5F   1F  0305    653         BLSSU   30$                     ; Branch if space exceeded
         5E   50   D0  0307    654         MOVL    R0,SP                   ; Allocate the space
                     030A    655
                     030A    656 ;
                     030A    657 ; Initialize the FAB and NAM blocks.
                     030A    658 ;
                     030A    659
              55   DD  030A    660         PUSHL   R5                      ; Save R5 across MOVC
```

```
20 AE   00B0 8F    00    6E    00    2C  030C   661          MOVC5   #0,(SP),#0,#<FAB$C_BLN+NAM$C_BLN>,FWA_FAB+4(SP) ; Clear FAB/NAM
                                  55 8ED0  0315   662          POPL    R5                              ; Restore R5
                      52    1C    AE    9E  0318   663          MOVAB   FWA_FAB(SP),R2                  ; Point to FAB
                      53    50    A2    9E  031C   664          MOVAB   FAB$C_BLN(R2),R3                ; Point to NAM
                      62  5003    8F    B0  0320   665          MOVW    #<FAB$C_BID!<FAB$C_BLN@8>>,FAB$B_BID(R2) ; Set FAB identifier
                      34    A2    55    90  0325   666          MOVB    R5,FAB$B_FNS(R2)                ; Set file name length
                      2C    A2    57    D0  0329   667          MOVL    R7,FAB$L_FNA(R2)                ; Set file name address
                      28    A2    63    9E  032D   668          MOVAB   (R3),FAB$L_NAM(R2)              ; Set NAM block address
                      63  6002    8F    B0  0331   669          MOVW    #<NAM$C_BID!<NAM$C_BLN@8>>,NAM$B_BID(R3) ; Set NAM identifier
                   0A A3    FF    8F    90  0336   670          MOVB    #NAM$C_MAXRSS,NAM$B_ESS(R3) ; Set ESA descriptor
                0C A3  00CC    CE    9E  033B   671          MOVAB   FWA_ESA(SP),NAM$L_ESA(R3)
                      56    27    9A  0341   672          MOVZBL  #SJC$_FILE_IDENTIFICATION,R6 ; Set up item code
                      57    14    A3    9E  0344   673          MOVAB   NAM$T_DVI(R3),R7                ; Point to DVI/FID/DID area
                                        0348   674
                                        0348   675 ;
                                        0348   676 ; Execute a PARSE and SEARCH to get the DVI/FID/DID.
                                        0348   677 ;
                                        0348   678
                                        0348   679          $PARSE  FAB=(R2)                        ; Parse the file name
                         0C    50    E9  0351   680          BLBC    R0,10$                          ; Branch if error
                                        0354   681          $SEARCH FAB=(R2)                        ; Search the file name
                         34    50    E8  035D   682          BLBS    R0,50$                          ; Branch to handle like FID item
                                        0360   683
                                        0360   684 ;
                                        0360   685 ; Helper branches.
                                        0360   686 ;
                                        0360   687
                       0223    31  0360   688 10$:     BRW     ERROR
                       FDA1    31  0363   689 20$:     BRW     BADPARAM
                       FDA3    31  0366   690 30$:     BRW     INSFMEM
                                   0369   691
                                   0369   692
                                   0369   693 FILE_IDENTIFICATION:                    ; Translate SJC$_FILE_IDENTIFICATION
                                   0369   694
                                   0369   695 ;
                                   0369   696 ;       R5  = buffer size
                                   0369   697 ;       R6  = item code
                                   0369   698 ;       R7  = buffer address
                                   0369   699 ;       R10 = pointer to lowest available stack address
                                   0369   700 ;
                                   0369   701 ; Check that the parameter is the correct length.  If it is not the expected
                                   0369   702 ; 28 bytes, and the previous mode is at least executive, assume that we have
                                   0369   703 ; been passed the entire expanded item and send it on as is.
                                   0369   704 ;
                                   0369   705
                      FC  AD    D5  0369   706          TSTL    FILE_ID(FP)                     ; See if there is already a filespec
                         F5    12  036C   707          BNEQ    20$                             ; Branch if so
                      1C    55    B1  036E   708          CMPW    R5,#28                          ; Ensure parameter is 28 bytes
                         0F    13  0371   709          BEQL    40$                             ; Branch if correct
                         50    DC  0373   710          MOVPSL  R0                              ; Get PSL
                50    50  02    16    EF  0375   711          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
                         01    50    D1  037A   712          CMPL    R0,#PSL$C_EXEC                  ; Previous mode exec or kernel?
                         E4    1A  037D   713          BGTRU   20$                             ; Branch if not
                      FE54    31  037F   714          BRW     INPUT_ITEM                      ; Branch to store item as is
                      0F    67    91  0382   715 40$:     CMPB    (R7),#15                        ; Ensure device no more than 15 bytes
                         DC    1A  0385   716          BGTRU   20$                             ; Branch if incorrect
                                   0387   717
```

```
                                0387   718 ;
                                0387   719 ; Check for sufficient space to allocate the work area, and do so.
                                0387   720 ;
                                0387   721
          50   FE35 CE    9E   0387   722         MOVAB   -FWA_SIZE(SP),R0        ; Get lowest address that will be used
               5A   50    D1   038C   723         CMPL    R0,RT0                  ; Compare against that available
                    D5    1F   038F   724         BLSSU   30$                     ; Branch if space exceeded
               5E   50    D0   0391   725         MOVL    R0,SP                   ; Allocate the space
                               0394   726
                               0394   727 ;
                               0394   728 ; Move the DVI/FID/DID to the work area.
                               0394   729 ;
                               0394   730
          6E   67    1C    28   0394   731 50$:    MOVC3   #28,(R7),FWA_DVI(SP)   ; Move the parameter to the work area
             FC AD    5E    D0   0398   732         MOVL    SP,FILE_ID(FP)          ; Save location of file ID buffer
               7E    56    3C   039C   733         MOVZWL  R6,-(SP)                ; Store item code, leave space for size
                  FDC0    31   039F   734         BRW     ITEM                    ; Remainder of processing comes later
                               03A2   735
                               03A2   736 ;
                               03A2   737 ; The file specification, if any, must be post-processed after all items
                               03A2   738 ; have been digested. Inputs:
                               03A2   739 ;
                               03A2   740 ;       R10 = address of file ID item
                               03A2   741 ;
                               03A2   742 ; Get a pointer to the DVI descriptor, and where the channel will be stored,
                               03A2   743 ; and initialize the descriptor.
                               03A2   744 ;
                               03A2   745
                               03A2   746 POSTPROCESS_FID:
2C AA  0040 8F   00   6E    00   2C   03A2   747         MOVC5   #0,(SP),#0,#FIB$C_LENGTH,FWA_FIB(R10) ; Initialize FIB
                         20   AA   D4   03AB   748         CLRL    32(R10)                 ; Clear unused longword
                    50   24   AA   9E   03AE   749         MOVAB   FWA_DVI_DESC(R10),R0    ; Point to DVI descriptor
                    53 0144   CA   9E   03B2   750         MOVAB   FWA_CHAN(R10),R3        ; Point to channel
                         60    6A   9A   03B7   751         MOVZBL  FWA_DVI(R10),(R0)       ; Store device name length
                 04 A0   01   AA   9E   03BA   752         MOVAB   FWA_DVI+1(R10),4(R0)    ; Store device name address
                               03BF   753
                               03BF   754 ;
                               03BF   755 ; Assign a channel to the device.
                               03BF   756 ;
                               03BF   757
                               03BF   758         $ASSIGN_S -                     ; Assign a channel
                               03BF   759                 DEVNAM=(R0), -          ;  Device name
                               03BF   760                 CHAN=(R3)               ;  Output channel number
                      91 50    E9   03CC   761         BLBC    R0,10$                  ; Branch if not assigned
                               03CF   762
                               03CF   763 ;
                               03CF   764 ; Build the FIB, the FIB descriptor, and the ACP attributes list.
                               03CF   765 ;
                               03CF   766
                    50   6C   AA   9E   03CF   767         MOVAB   FWA_ATRLIST(R10),R0     ; Point to attribute list
                    51   24   AA   9E   03D3   768         MOVAB   FWA_FIB_DESC(R10),R1    ; Point to FIB descriptor
                    52 0148   CA   9E   03D7   769         MOVAB   FWA_IOSB(R10),R2        ; Point to IOSB
                    54   2C   AA   9E   03DC   770         MOVAB   FWA_FIB(R10),R4         ; Point to FIB
                               03E0   771
                 04 A4   10   AA   D0   03E0   772         MOVL    FWA_FID(R10),FIB$W_FID(R4) ; Store file ID
                 08 A4   14   AA   B0   03E5   773         MOVW    FWA_FID+4(R10),FIB$W_FID+4(R4)
                 18 F8 AD    00   E1   03EA   774         BBC     #DELETE_FLAG,FLAGS(FP),55$ ; Branch if not deleting file
```

```
                OA A4    16 AA    D0  03EF    775          MOVL    FWA_DID(R10),FIB$W_DID(R4) ; Also store directory ID
                OE A4    1A AA    B0  03F4    776          MOVW    FWA_DID+4(R10),FIB$W_DID+4(R4)
             14 A4      0800 8F    A8  03F9    777          BISW    #FIB$M_FINDFID,FIB$W_NMCTL(R4)
                38 A4     01 C8    C8  03FF    778          BISL    #FIB$M_ALT_REQ,FIB$L_STATUS(R4) ; Alternate access required
                3C A4     08 D0    D0  0403    779          MOVL    #ARM$M_DELETE,FIB$L_ALT_ACCESS(R4) ; Check for delete access
                                        0407    780
             61    00000040 8F    D0  0407    781 55$:     MOVL    #FIB$C_LENGTH,(R1)      ; Initialize FIB descriptor
                   04 A1     64 9E    040E    782          MOVAB   (R4),4(R1)              ;
                                        0412    783
             60    00040020 8F    D0  0412    784          MOVL    #<ATR$S_RECATTR+<ATR$C_RECATTR@16>>,(R0)
               04 A0     0124 CA    9E  0419    785          MOVAB   FWA_RECATTR(R10),4(R0)
            08 A0     002E0100 8F    D0  041F    786          MOVL    #<256+<ATR$C_FILE_SPEC@16>>,8(R0)
                0C A0     24 AA    9E  0427    787          MOVAB   FWA_FILE_SPEC(R10),12(R0)
                      10 A0    D4  042C    788          CLRL    16(R0)
                                        042F    789
                                        042F    790 ;
                                        042F    791 ; Access the file to get necessary information.
                                        042F    792 ;
                                        042F    793
                                        042F    794          $QIOW_S -                       ; Issue QIO to obtain file attributes
                                        042F    795                  EFN=EFN(AP), -          ;  User's event flag
                                        042F    796                  CHAN=(R3), -            ;  Channel number
                                        042F    797                  FUNC=#IO$_ACCESS, -     ;  Read attributes function code
                                        042F    798                  IOSB=(R2), -            ;  I/O status block
                                        042F    799                  P1=(R1), -              ;  Address of FIB descriptor
                                        042F    800                  P5=R0                   ;  Address of attribute list
                         50    DD  044D    801          PUSHL   R0                      ; Save $QIOW status
                                        044F    802          $DASSGN_S -                     ; Deassign the channel
                                        044F    803                  CHAN=(R3)               ;  Channel number
                         50 8ED0  0459    804          POPL    R0                      ; Restore status from access
                      48 50    E9  045C    805          BLBC    R0,70$                  ; Branch if $QIOW failed
                50    0148 CA    3C  045F    806          MOVZWL  FWA_IOSB(R10),R0        ; Pick up status from IOSB
                      40 50    E9  0464    807          BLBC    R0,70$                  ; Branch if operation failed
                                        0467    808
                                        0467    809 ;
                                        0467    810 ; Compute the file size from the record attributes.
                                        0467    811 ;
                                        0467    812
          1C AA    012C CA    10  9C  0467    813          ROTL    #16, -                  ; Move EFBLK to file size area and
                                        046E    814                  FWA_RECATTR+FAT$L_EFBLK(R10), -
                                        046E    815                  FWA_FILE_SIZE(R10)      ; convert to unswapped
                         09 13    046E    816          BEQL    60$                     ; Branch if EFBLK is zero
                      0130 CA    B5  0470    817          TSTW    FWA_RECATTR+FAT$W_FFBYTE(R10) ; Test first free byte
                         03 12    0474    818          BNEQ    60$                     ; Branch if nonzero
                      1C AA    D7  0476    819          DECL    FWA_FILE_SIZE(R10)      ; Adjust EFBLK
                                        0479    820
                                        0479    821 ;
                                        0479    822 ; Slide the real data up adjacent to the previous item on the stack, and
                                        0479    823 ; finish it by adding the length and item code.
                                        0479    824 ;
                                        0479    825
                   57    24 AA    3C  0479    826 60$:     MOVZWL  FWA_FILE_SPEC(R10),R7   ; Get file specification length
                00FE 8F      57 B1  047D    827          CMPW    R7,#254                 ; Check against maximum supported length
                         05 1B    0482    828          BLEQU   65$                     ; Branch if in range
                57    00FE 8F    3C  0484    829          MOVZWL  #254,R7                 ; Shorten to maximum
                   57    26 C0    0489    830 65$:     ADDL    #FWA_FILE_SPEC+2,R7     ; Add fixed portion
                     FE AA    57 B0  048C    831          MOVW    R7,-2(R10)              ; Store length in message
```

```
                58  000001CB 8F    57    C3  0490    832          SUBL3   R7,#FWA_SIZE,R8         ; Compute bias
                           57    5A    C0  0498    833          ADDL    R10,R7                  ; Compute size of area above filespec
                           57    5E    C2  049B    834          SUBL    SP,R7                   ; = R10 - SP + R7
                     6E48  6E    57    28  049E    835          MOVC3   R7,(SP),(SP)[R8]        ; Squish out unused space
                           5E    58    C0  04A3    836          ADDL    R8,SP                   ; Delete unused stack
                                       05  04A6    837          RSB                             ; Done with file spec
                                           04A7    838
                                           04A7    839 ;
                                           04A7    840 ; Helper branches.
                                           04A7    841 ;
                                           04A7    842
                                 00DC      31  04A7    843 70$:     BRW     ERROR
                                           04AA    844
                                           04AA    845          .DSABL  LSB
```

```
                                04AA   847
                                04AA   848 ;
                                04AA   849 ; Stack work area offsets for next routine.
                                04AA   850 ;
                                04AA   851
                00000000        04AA   852 LWA_BUFFER=     0                       ; Logical name buffer
                00000100        04AA   853 LWA_LOGNAM=     256                     ; Logical name descriptor
                00000108        04AA   854 LWA_ITMLST=     264                     ; $TRNLNM item list
                00000124        04AA   855 LWA_RSLLEN=     292                     ; Translation length buffer
                00000128        04AA   856 LWA_ATTRBUF=    296                     ; Translation attributes buffer
                0000012C        04AA   857 LWA_SIZE=       300                     ; Work area length
                                04AA   858
                                04AA   859             .ENABL  LSB
                        FC5A    04AA   860 10$:     BRW     BADPARAM
                        FC5C    04AD   861 20$:     BRW     INSFMEM
                                04B0   862
                                04B0   863 TRANSLATE_OBJECT:                    ; Translate object names
                                04B0   864
                                04B0   865 ;
                                04B0   866 ;       R5  = buffer size
                                04B0   867 ;       R6  = item code
                                04B0   868 ;       R7  = buffer address
                                04B0   869 ;       R10 = pointer to lowest available stack address
                                04B0   870 ;
                                04B0   871 ; Check that the parameter is the correct length and that there is sufficient
                                04B0   872 ; space to allocate the work area (then do so).
                                04B0   873 ;
                                04B0   874
                00FF    8F  55  B1  04B0   875             CMPW    R5,#255             ; Ensure no more than 255 bytes
                            F3  1A  04B5   876             BGTRU   10$                 ; Branch if incorrect
                                04B7   877
                53      FED4 CE     9E  04B7   878             MOVAB   -LWA_SIZE(SP),R3    ; Get lowest address that will be used
                        5A      53  D1  04BC   879             CMPL    R3,RTO              ; Compare against that available
                                EC  1F  04BF   880             BLSSU   20$                 ; Branch if space exceeded
                        5E      53  D0  04C1   881             MOVL    R3,SP               ; Allocate the space
                                04C4   882
                                04C4   883 ;
                                04C4   884 ; Prepare to perform the iterative translations by initializing the logical name
                                04C4   885 ; descriptor and the item list utilized by the recursive $TRNLNMs.
                                04C4   886 ;
                                04C4   887
                                55  DD  04C4   888             PUSHL   R5                  ; Save the input string length
                        63      67  28  04C6   889             MOVC3   R5,(R7),(R3)        ; Move input string into scratch buffer
                                55 8ED0 04CA   890             POPL    R5                  ; Restore the input string length
                                57  5E  04CD   891             MOVL    SP,R7               ; Restore scratch buffer address
                                04D0   892
                51      0104 C7     9E  04D0   893             MOVAB   LWA_LOGNAM+4(R7),R1 ; Addr of area requiring initialization
                        81      67  9E  04D5   894             MOVAB   (R7),(R1)+          ; Init log name descriptor buffer addr
                                04D8   895
            81   000200FF  8F      D0  04D8   896             MOVL    #<LNM$_STRING @ 16+-    ; Init string item list item type
                                04DF   897                             255>,(R1)+          ;  and string buffer length
                        81      67  9E  04DF   898             MOVAB   (R7),(R1)+          ; Init string item buffer address
                    81  0124 C7     9E  04E2   899             MOVAB   LWA_RSLLEN(R7),(R1)+ ; Init string item return buffer address
            81   00030004  8F      D0  04E7   900             MOVL    #<LNM$_ATTRIBUTES @ 16+-; Init attributes item list item type
                                04EE   901                             4>,(R1)+            ;  and attributes buffer length
                    81  0128 C7     9E  04EE   902             MOVAB   LWA_ATTRBUF(R7),(R1)+ ; Init attributes item buffer address
                                61  7C  04F3   903             CLRQ    (R1)                ; Init attributes item return buffer
```

```
                    04F5  904                                     ; address and end of item list marker
                    04F5  905
54  0A  D0  04F5  906       MOVL    #LNM$C_MAXDEPTH,R4          ; Initialize loop counter
```

```
                     04F8   908
                     04F8   909 ;
                     04F8   910 ; Loop to iterate over translations.
                     04F8   911 ;
                     04F8   912 ;       R4  = Current translation count
                     04F8   913 ;       R5  = Current input string length
                     04F8   914 ;       R6  = Item code
                     04F8   915 ;       R7  = Current input string address, and
                     04F8   916 ;             Address of work area
                     04F8   917 ;
                     04F8   918 ; The iterations successfully terminate when:
                     04F8   919 ;
                     04F8   920 ; 1. The maximum translation recursion depth is exceeded.
                     04F8   921 ; 2. The current translation succeeds but the translation is marked terminal.
                     04F8   922 ; 3. The current translation fails with an error of SS$_NOLOGNAM.
                     04F8   923 ;
                     04F8   924 ; The iterations unsuccessfully terminate when:
                     04F8   925 ;
                     04F8   926 ; 1. The current translation fails with some error other than SS$_NOLOGNAM.
                     04F8   927 ; 2. The current translation exceeds but the equivalence string is either of
                     04F8   928 ;    null length or does not exist.
                     04F8   929 ;
                     04F8   930
  0100 C7    55  D0  04F8   931 30$:    MOVL    R5,LWA_LOGNAM(R7)          ; Store name length in descriptor
                     04FD   932
                     04FD   933         $TRNLNM_S       -                  ; Attempt to translate the name
                     04FD   934                 ATTR    = TRNLNM_ATTR,-          ; Case-insensitive translation
                     04FD   935                 ITMLST  = LWA_ITMLST(R7),-       ; Address of item list
                     04FD   936                 LOGNAM  = LWA_LOGNAM(R7),-       ; Address of name descriptor
                     04FD   937                 TABNAM  = TRNLNM_TABLE           ; Addr LNM$FILE_DEV descriptor
        19 50    E9  0516   938         BLBC    R0,45$                     ; Done if translation fails
                     0519   939
  55    0124 C7    3C  0519   940         MOVZWL  LWA_RSLLEN(R7),R5          ; Retrieve length of equivalence string
               0B    13  051E   941         BEQL    40$                        ; Go return error if length is 0
                     0520   942                                            ; (null or non-existant translation)
                     0520   943
               09    E0  0520   944         BBS     #LNM$V_TERMINAL,-          ; Is the translation marked terminal?
    13 0128 C7             0522   945                 LWA_ATTRBUF(R7),50$        ; terminate iterative translations if so
          CF   54    F5  0526   946         SOBGTR  R4,30$                     ; Continue if more translations possible
               0E    11  0529   947         BRB     50$                        ; Else, done if xlation count exhausted
                     052B   948
  50    0154 8F    3C  052B   949 40$:    MOVZWL  #SS$_IVLOGNAM,R0           ; Return an error for null length or
               54    11  0530   950         BRB     ERROR                      ; non-existant translation
                     0532   951
  01BC 8F    50  B1  0532   952 45$:    CMPW    R0,#SS$_NOLOGNAM           ; If the translation failed for a reason
               4D    12  0537   953         BNEQ    ERROR                      ; other than the logical name did not
                     0539   954                                            ; exist then go return the error
```

```
                                0539   956
                                0539   957 ;
                                0539   958 ; Recursive translations have completed. Format the final translation by
                                0539   959 ; removing blanks, tabs, null characters, and a trailing colon if there is one,
                                0539   960 ; and upcasing the name using the DEC multi-national character upcasing table.
                                0539   961 ;
                                0539   962 ;       R1 = Current character
                                0539   963 ;       R2 = Current character index
                                0539   964 ;       R3 = Cursor to output buffer
                                0539   965 ;       R5 = Length of input string
                                0539   966 ;       R7 = Address of input string
                                0539   967 ;
                                0539   968
            52    01    CE      0539   969 50$:    MNEGL   #1,R2                   ; Initialize the loop index
            53    57    D0      053C   970         MOVL    R7,R3                   ; Initialize output buffer cursor
                  18    11      053F   971         BRB     70$                     ; Branch to enter the loop
                                0541   972
          51    6742    9A      0541   973 60$:    MOVZBL  (R7)[R2],R1             ; Pick up the current character
                  12    13      0545   974         BEQL    70$                     ; Remove it if it is null
            20    51    91      0547   975         CMPB    R1,#^A' '               ; Is the current character a blank?
                  0D    13      054A   976         BEQL    70$                     ; Remove it if it is
            09    51    91      054C   977         CMPB    R1,#^O011               ; Is the current character a tab?
                  08    13      054F   978         BEQL    70$                     ; Remove it if it is
    83  00000000'GF41    90      0551   979         MOVB    G^EXE$UPCASE_DAT[R1],-  ; Move upcased character into output
                                0559   980         (R3)+                           ; buffer
          E4 52    55    F2      0559   981 70$:    AOBLSS  R5,R2,60$               ; Continue loop until done
                                055D   982
          57    53    57    C3      055D   983         SUBL3   R7,R3,R7                ; Computes name's compressed length
                  C8    13      0561   984         BEQL    40$                     ; Return an error if its zero
            3A    FF    A3      0563   985         CMPB    -1(R3),#^A':'           ; Is there a trailing colon?
                  04    12      0567   986         BNEQ    80$                     ; Branch if there isn't
                  57    D7      0569   987         DECL    R7                      ; Otherwise remove it
                  BE    13      056B   988         BEQL    40$                     ; Return an error if name length is 0
                                056D   989
                                056D   990 ;
                                056D   991 ; Slide the name up the stack so that it is adjacent to the previous item on
                                056D   992 ; the stack. Then complete the formation of the item by adding the name length
                                056D   993 ; and item code.
                                056D   994 ;
                                056D   995
    58  0000012C    8F    57    C3      056D   996 80$:    SUBL3   R7,#LWA_SIZE,R8          ; Compute bias
          6E48      6E    57    28      0575   997         MOVC3   R7,(SP),(SP)[R8]        ; Slide item up
                    5E    58    C0      057A   998         ADDL2   R8,SP                   ; Delete unused stack
                    7E    57    B0      057D   999         MOVW    R7,-(SP)                ; Store item length
                    7E    56    B0      0580  1000         MOVW    R6,-(SP)                ; Store item code
                        FBDC    31      0583  1001         BRW     ITEM                    ; Return to item list processing
                                0586  1002         .DSABL  LSB
```

```
                                0586  1004
                                0586  1005 ;
                                0586  1006 ; Synchronous error return path.  Store status in the IOSB, set the event flag,
                                0586  1007 ; and declare the AST, if specified.
                                0586  1008 ;
                                0586  1009
                     50   DD    0586  1010 ERROR:   PUSHL   R0                              ; Save completion status
          00000000'GF     6C    0588  1011          CALLG   (AP),G^SYS$SETEF               ; Set specified event flag
                51   14 AC    058F  1012          MOVL    IOSB(AP),R1                    ; Get address of IOSB
                     09   13  0593  1013          BEQL    10$                            ; Branch if none
                                0595  1014          IFNOWRT #8,(R1),10$                    ; Branch if no write access
                61      6E    DO    059B  1015          MOVL    (SP),(R1)                      ; Store completion status
                51   18 AC    DO    059E  1016 10$:     MOVL    ASTADR(AP),R1                  ; Get address of AST routine
                     15   13  05A2  1017          BEQL    20$                            ; Branch if none
                     50   DC  05A4  1018          MOVPSL  R0                             ; Get PSL
50   50   02     16   EF    05A6  1019          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R0,R0 ; Get previous mode
                                05AB  1020          $DCLAST_S (R1),ASTPRM(AP),R0           ; Declare completion AST
                     50 8ED0  05B9  1021 20$:     POPL    R0                             ; Restore completion status
                        04    05BC  1022          RET                                    ; Return with error status
```

```
                                                M 12
05BD  1024          .SBTTL  EXE$JBCRSP - Store response from job controller
05BD  1025
05BD  1026 ;++
05BD  1027 ;
05BD  1028 ; EXE$JBCRSP - STORE RESPONSE FROM JOB CONTROLLER
05BD  1029 ;
05BD  1030 ; FUNCTIONAL DESCRIPTION:
05BD  1031 ;
05BD  1032 ;       This routine is called as a special kernel AST routine to return status
05BD  1033 ;       from the send to job controller system service to the requesting
05BD  1034 ;       process.  It ensures that the same image is executing and then sets the
05BD  1035 ;       specified event flag, stores a status value in the IOSB if specified,
05BD  1036 ;       stores data in any output buffer items that were in the original
05BD  1037 ;       request, and declares the completion AST if specified.  If appropriate,
05BD  1038 ;       the ACB is deallocated.
05BD  1039 ;
05BD  1040 ; INPUTS:
05BD  1041 ;       R0-R3           = scratch
05BD  1042 ;       R4              = PCB address
05BD  1043 ;       R5              = ACB address
05BD  1044 ;
05BD  1045 ; OUTPUTS:
05BD  1046 ;       See above.
05BD  1047 ;--
05BD  1048
05BD  1049 EXE$JBCRSP::                                ; Response from job controller
05BD  1050
05BD  1051 ;
05BD  1052 ; Compare the image count when the request was queued with the current image
05BD  1053 ; count.  If different, a new image is running - do not store anything.
05BD  1054 ;
05BD  1055
53   00000000'9F    D0  05BD  1056          MOVL    @#CTL$GL_PHD,R3             ; Get PHD address
 1C A5    00F4 C3    D1  05C4  1057          CMPL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R5) ; See if image count correct
               03    13  05CA  1058          BEQL    10$                         ; Branch if correct
             0085    31  05CC  1059          BRW     70$                         ; Join code to deallocate ACB
                     05CF  1060
                     05CF  1061 ;
                     05CF  1062 ; Loop over the return item descriptors storing information in the user's
                     05CF  1063 ; output buffers.  During this loop:
                     05CF  1064 ;
                     05CF  1065 ;       R5  = user buffer address
                     05CF  1066 ;       R6  = pointer to item descriptors in ACB
                     05CF  1067 ;       R7  = user buffer size
                     05CF  1068 ;       R8  = actual data size
                     05CF  1069 ;       R9  = requester's access mode
                     05CF  1070 ;       R10 = item count
                     05CF  1071 ;
                     05CF  1072
          2C A5      B5  05CF  1073 10$:     TSTW    ACB_W_ITEMCOUNT(R5)         ; Any items to return?
               52    13  05D2  1074          BEQL    50$                         ; Branch if none
       07F0 8F       BB  05D4  1075          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10>  ; Save registers
    56    2E A5      9E  05D8  1076          MOVAB   ACB_B_ITEMS(R5),R6          ; Point to items
    59    0B A5      9A  05DC  1077          MOVZBL  ACB$B_RMOD(R5),R9           ; Get requester's mode
    5A    2C A5      3C  05E0  1078          MOVZWL  ACB_W_ITEMCOUNT(R5),R10     ; Get item count
       57    86      3C  05E4  1079 20$:     MOVZWL  (R6)+,R7                    ; Get user buffer size
       58    86      3C  05E7  1080          MOVZWL  (R6)+,R8                    ; Get actual size
```

```
                        55    86    D0   05EA  1081          MOVL    (R6)+,R5                        ; Get data buffer address
                        50    55    D0   05ED  1082          MOVL    R5,R0                           ; R0 = buffer address
                        51    57    D0   05F0  1083          MOVL    R7,R1                           ; R1 = buffer length
                        53    59    D0   05F3  1084          MOVL    R9,R3                           ; R3 = requester's mode
                00000000'EF   16    05F6  1085          JSB     EXE$PROBEW                      ; Probe for write access
                        66    50    E9   05FC  1086          BLBC    R0,90$                          ; Branch if inaccessible
    65  57  00  04 A6      58    2C   05FF  1087          MOVC5   R8,4(R6),#0,R7,(R5)             ; Move data to user buffer
                        55    86    D0   0606  1088          MOVL    (R6)+,R5                        ; Get return length address
                              11    13   0609  1089          BEQL    40$                             ; Branch if none
                                    060B  1090          IFNOWRT #2,(R5),90$,R9                  ; Probe for write access
                        57    58    D1   0611  1091          CMPL    R8,R7                           ; Minimize user and actual length
                              03    1E   0614  1092          BGEQU   30$                             ; Branch if actual length larger
                        57    58    D0   0616  1093          MOVL    R8,R7                           ; Get actual length as minimum
                        65    57    B0   0619  1094 30$:     MOVW    R7,(R5)                         ; Return buffer length
                        56    58    C0   061C  1095 40$:     ADDL2   R8,R6                           ; Advance over data
                           C2 5A    F5   061F  1096          SOBGTR  R10,20$                         ; Loop for all items
                     07F0 8F        BA   0622  1097          POPR    #^M<R4,R5,R6,R7,R8,R9,R10> ; Restore registers
                                         0626  1098
                                         0626  1099 ;
                                         0626  1100 ; Output buffers stored.  Set the specified event flag, return status to the
                                         0626  1101 ; IOSB, and declare the completion AST if specified.  If no AST specified,
                                         0626  1102 ; deallocate the ACB.
                                         0626  1103 ;
                                         0626  1104
                    51    60 A4    D0   0626  1105 50$:     MOVL    PCB$L_PID(R4),R1                ; R1 = PID
                                    52   D4   062A  1106          CLRL    R2                              ; R2 = null priority increment
                    53    20 A5    9A   062C  1107          MOVZBL  ACB_L_EFN(R5),R3                ; R3 = event flag number
                00000000'EF   16    0630  1108          JSB     SCH$POSTEF                      ; Set specified event flag
                    51    24 A5    D0   0636  1109          MOVL    ACB_L_IOSB(R5),R1               ; Get IOSB address
                              0B    13   063A  1110          BEQL    60$                             ; Branch if none
                                    063C  1111          IFNOWRT #4,(R1),60$,ACB$B_RMOD(R5) ; Probe for write access
                    61    28 A5    D0   0643  1112          MOVL    ACB_L_STATUS(R5),(R1)           ; Return status
                        10 A5    D5   0647  1113 60$:     TSTL    ACB$L_AST(R5)                   ; Completion AST specified?
                              08    13   064A  1114          BEQL    70$                             ; Branch if no to deallocate ACB
                                    52   D4   064C  1115          CLRL    R2                              ; R2 = null priority increment
                00000000'EF   17    064E  1116          JMP     SCH$QAST                        ; Queue completion AST and return
                                         0654  1117
                                         0654  1118 ;
                                         0654  1119 ; Processing finished.  Return AST quota if charged, and deallocate the ACB.
                                         0654  1120 ; (No byte count quota is charged for this ACB because it is allocated by the
                                         0654  1121 ; job controller.)
                                         0654  1122 ;
                                         0654  1123
03 0B A5                06    E1   0654  1124 70$:     BBC     #ACB$V_QUOTA,ACB$B_RMOD(R5),80$ ; Branch if no AST quota charged
                        38 A4    B6   0659  1125          INCW    PCB$W_ASTCNT(R4)                ; Return AST quota
                    50    55    D0   065C  1126 80$:     MOVL    R5,R0                           ; R0 = ACB address
                00000000'EF   17    065F  1127          JMP     EXE$DEANONPAGED                 ; Deallocate ACB and return
                                         0665  1128
                                         0665  1129 ;
                                         0665  1130 ; Memory is inaccessible.  Attempt to return an access violation status to
                                         0665  1131 ; the IOSB.
                                         0665  1132 ;
                                         0665  1133
                     07F0 8F        BA   0665  1134 90$:     POPR    #^M<R4,R5,R6,R7,R8,R9,R10> ; Restore registers
                28 A5    0C    D0   0669  1135          MOVL    #SS$_ACCVIO,ACB_L_STATUS(R5) ; Force status to ACCVIO
                              B7    11   066D  1136          BRB     50$                             ; Go to return EFN and IOSB
                                    066F  1137          .END
```

```
$$.TMP1                 =  00000001
$$.TMP2                 =  00000062
$$T1                    =  00000000
ACB$B_RMOD              =  0000000B
ACB$L_AST               =  00000010
ACB$L_KAST              =  00000018
ACB$V_QUOTA             =  00000006
ACB_B_ITEMS                0000002E
ACB_L_EFN                  00000020
ACB_L_IMGCNT               0000001C
ACB_L_IOSB                 00000024
ACB_L_STATUS               00000028
ACB_W_ITEMCOUNT            0000002C
ACCVIO                     00000102 R      02
ACM$Q_SYSTIME           =  0000003C
ARM$M_DELETE            =  00000008
ASTADR                  =  00000018
ASTPRM                  =  0000001C
ATR$C_FILE_SPEC         =  0000002E
ATR$C_RECATTR           =  00000004
ATR$S_RECATTR           =  00000020
BADPARAM                   00000107 R      02
BOOL_ITEM               =  0000000C
CTL$AL_STACKLIM            ********   X    02
CTL$GL_PCB                 ********   X    02
CTL$GL_PHD                 ********   X    02
CTL$T_USERNAME             ********   X    02
DELETE_FLAG             =  00000000
EFN                     =  00000004
ERROR                      00000586 R      02
EXE$DEANONPAGED            ********     X  02
EXE$GETQUI                 00000114 RG     02
EXE$GQ_SYSTIME             ********     X  02
EXE$JBCRSP                 000005BD RG     02
EXE$PROBER                 ********     X  02
EXE$PROBEW                 ********     X  02
EXE$SENDMSG                ********     X  02
EXE$SNDJBC                 0000011C RG     02
EXE$UPCASE_DAT             ********     X  02
FAB$B_BID               =  00000000
FAB$B_FNS               =  00000034
FAB$C_BID               =  00000003
FAB$C_BLN               =  00000050
FAB$L_FNA               =  0000002C
FAB$L_NAM               =  00000028
FAT$L_EFBLK             =  00000008
FAT$W_FFBYTE            =  0000000C
FIB$C_LENGTH            =  00000040
FIB$L_ALT_ACCESS        =  0000003C
FIB$L_STATUS            =  00000038
FIB$M_ALT_REQ           =  00000001
FIB$M_FINDFID           =  00000800
FIB$W_DID               =  0000000A
FIB$W_FID               =  00000004
FIB$W_NMCTL             =  00000014
FILE_ID                 =  FFFFFFFC
FILE_IDENTIFICATION        00000369 R      02
FILE_SPECIFICATION         000002F1 R      02
FINISH_MESSAGE             0000021D R      02
FIXED_AREA              =  0000005A
FLAGS                   =  FFFFFFF8
FUNC                    =  00000008
FWA_ATRLIST             =  0000006C
FWA_CHAN                =  00000144
FWA_DID                 =  00000016
FWA_DVI                 =  00000000
FWA_DVI_DESC            =  00000024
FWA_ESA                 =  000000CC
FWA_FAB                 =  0000001C
FWA_FIB                 =  0000002C
FWA_FIB_DESC            =  00000024
FWA_FID                 =  00000010
FWA_FILE_SIZE           =  0000001C
FWA_FILE_SPEC           =  00000024
FWA_IOSB                =  00000148
FWA_NAM                 =  0000006C
FWA_RECATTR             =  00000124
FWA_SIZE                =  000001CB
GETQUI_BOOL_ITEM           00000020 R      02
GETQUI_DATA                000000D2 R      02
GETQUI_OUTPUT_ITEM         00000060 R      02
GETQUI_SPECIAL_TABLE       000000B2 R      02
INPUT_ITEM                 000001D6 R      02
INSFMEM                    0000010C R      02
IO$_ACCESS              =  00000032
IOSB                    =  00000014
ITEM                       00000162 R      02
ITMLST                  =  00000010
LNM$C_MAXDEPTH          =  0000000A
LNM$M_CASE_BLIND        =  02000000
LNM$V_TERMINAL          =  00000009
LNM$_ATTRIBUTES         =  00000003
LNM$_STRING             =  00000002
LWA_ATTRBUF             =  00000128
LWA_BUFFER              =  00000000
LWA_ITMLST              =  00000108
LWA_LOGNAM              =  00000100
LWA_RSLLEN              =  00000124
LWA_SIZE                =  0000012C
MAX_FUNC                =  00000004
MAX_ITEM                =  00000008
MSG$_GETQUI             =  00000010
MSG$_SNDJBC             =  0000000F
MSG_CODE                =  00000000
NAM$B_BID               =  00000000
NAM$B_ESS               =  0000000A
NAM$C_BID               =  00000002
NAM$C_BLN               =  00000060
NAM$C_MAXRSS            =  000000FF
NAM$L_ESA               =  0000000C
NAM$T_DVI               =  00000014
NULARG                  =  0000000C
OUTPUT_ITEM             =  00000010
PCB$B_PRIB              =  0000002F
```

```
PCB$L_EOWNER                    = 00000068
PCB$L_EPID                      = 00000064
PCB$L_PID                       = 00000060
PCB$L_STS                       = 00000024
PCB$L_UIC                       = 000000BC
PCB$T_TERMINAL                  = 00000044
PCB$W_ASTCNT                    = 00000038
PHD$L_IMGCNT                    = 000000F4
PHD$Q_PRIVMSK                   = 00000000
POSTPROCESS_FID                   000003A2 R      02
PSL$C_EXEC                      = 00000001
PSL$S_CURMOD                    = 00000002
PSL$S_PRVMOD                    = 00000002
PSL$V_CURMOD                    = 00000018
PSL$V_PRVMOD                    = 00000016
QUI$_RESERVED_FUNC_2            = 00000009
QUI$_RESERVED_OUTPUT_6          = 0000005E
QUI$_SEARCH_NAME                = 0000004D
SCH$CLREF                         ********    X   02
SCH$POSTEF                        ********    X   02
SCH$QAST                          ********    X   02
SJC$_CHARACTERISTIC_NAME        = 0000000C
SJC$_DELETE_FILE                = 00C00018
SJC$_DESTINATION_QUEUE          = 0000001A
SJC$_FILE_IDENTIFICATION        = 00000027
SJC$_FILE_SPECIFICATION         = 0000002A
SJC$_FORM_NAME                  = 00000036
SJC$_GENERIC_TARGET             = 00000046
SJC$_LOG_QUEUE                  = 00000061
SJC$_QUEUE                      = 00000086
SJC$_RESERVED_FUNC_2            = 00000020
SJC$_RESERVED_OUTPUT_2          = 000000AB
SNDJBC_BOOL_ITEM                  00000000 R      02
SNDJBC_DATA                       000000EA R      02
SNDJBC_OUTPUT_ITEM                00000040 R      02
SNDJBC_SPECIAL_TABLE              00000080 R      02
SPECIAL_TABLE                   = 00000014
SS$_ACCVIO                      = 0000000C
SS$_BADPARAM                    = 00000014
SS$_EXASTLM                     = 00002A04
SS$_INSFMEM                     = 00000124
SS$_IVLOGNAM                    = 00000154
SS$_NOLOGNAM                    = 000001BC
SYS$ASSIGN                        ********   GX   02
SYS$CMKRNL                        ********   GX   02
SYS$DASSGN                        ********   GX   02
SYS$DCLAST                        ********   GX   02
SYS$GL_JOBCTLMB                   ********    X   02
SYS$PARSE                         ********   GX   02
SYS$QIOW                          ********   GX   02
SYS$SEARCH                        ********   GX   02
SYS$SETEF                         ********    X   02
SYS$TRNLNM                        ********   GX   02
TRANSLATE_OBJECT                  000004B0 R      02
TRNLNM_ATTR                       000000BA R      02
TRNLNM_TABLE                      000000BE R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	0000002E (46.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	0000066F (1647.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.26
Command processing	110	00:00:00.54	00:00:01.22
Pass 1	566	00:00:23.66	00:00:27.07
Symbol table sort	0	00:00:03.96	00:00:04.13
Pass 2	208	00:00:04.77	00:00:05.38
Symbol table output	20	00:00:00.18	00:00:00.20
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	938	00:00:33.22	00:00:38.29

The working set limit was 1800 pages.
135859 bytes (266 pages) of virtual memory were used to buffer the intermediate code.
There were 140 pages of symbol table space allocated to hold 2520 non-local and 48 local symbols.
1137 source lines were read in Pass 1, producing 17 object records in Pass 2.
40 pages of virtual memory were used to define 38 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	7
_255DUA28:[SYSLIB]STARLET.MLB;2	28
TOTALS (all libraries)	35

2706 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:SYSSNDJBC/OBJ=OBJ$:SYSSNDJBC MSRC$:SYSSNDJBC/UPDATE=(ENH$:SYSSNDJBC)+EXECML$/LIB
```

SYSSETPRA LIS
SYSSETSSF LIS
SYSSNDJBC LIS
SYSSETIME LIS
SYSSETPFM LIS
SYSUPDSEC LIS
SYSSNDMSG LIS
SYSSETPRT LIS
SYSSETPRV LIS
SYSUNWIND LIS
SYSSETSTK LIS
SYSSETMOD LIS
SYSSETEXV LIS
SYSSETPRI LIS
SYSSCHEVT LIS